TOWARD A COMPUTATIONAL THEORY FOR MOTION UNDERSTANDING: THE EXPERT ANIMATORS MODEL

Ahmed S. MOHAMED
William W. ARMSTRONG

Department of Computing Science
The University of Alberta
Edmonton, Alberta, Canada T6G 2H1

ABSTRACT

Artificial intelligence researchers claim to "understand" some aspect of human intelligence when their model is able to "emulate" it. In the context of computer graphics, the ability to go from motion representation to "convincing" animation should accordingly be treated not simply as a trick for computer graphics programmers but as important epistemological and methodological goal. In this paper we investigate a unifying model for animating a group of articulated bodies such as humans and robots in a three-dimensional environment. The proposed model is considered in the framework of knowledge representation and processing , with special reference to "motion" knowledge. The model is meant to help setting the basis for a computational theory for motion understanding applied to articulated bodies.

[1] INTRODUCTION

Articulated body movements cannot be understood by using the current techniques of computer animation even if they incorporate dynamics to model every detail. There is a significant difference between "understanding" the motion and "synthesizing" it. This is the difference between the point of view of "cybernetics" and the point of view of "computer graphics". The latter starts from a set of trajectories or forces and torques acting on a body, and tries to account for the resulting movement: "how is the arm going to move if the hand has to follow such a trajectory?", "how is the body going to react if the legs are exerting such a force on the ground?".

The point of view of cybernetics is just the opposite: it starts from the definition of the goal (a desired behaviour) and then tries to force the system to follow it, possibly using a large range of current motion control techniques, either of the feedback or feedforward type. The emphasis, in this case, is on the knowledge that is required to produce fluent natural motion performance: sensory knowledge, motor knowledge, knowledge processing, abstract representation of knowledge structures. These are all incorporated in our term "motion" knowledge.

In AI terminology, knowledge representation means defining concepts and rules able to capture the essential complexity of a given problem domain that escapes a direct approach [20]. Thus, a knowledge representation approach emphasizes the "goals" and the "functions" of a system rather than its specific mechanisms, and in so doing it generalizes and abstracts our understanding of the system, since it is likely that different mechanisms can be found that can implement the same function and goal.

Although knowledge representation and processing are "hot" topics in artificial intelligence research, very few attempts have been made to help understand motion, which means to build up a model of "motion" knowledge. In this paper, we concentrate on the knowledge aspects of motion and propose a unifying model for animating a group of articulated bodies that we hope will be useful for investigators who are dealing with articulated body animation in the large. This includes neuroscientists, biomechanists, dance designers, motor rehabilitators, computer animators, and robotics researchers. It is believed that all these researchers basically require

the same type of motion knowledge whether they are writing a computer script for dance ballet, a program for figure animation, a control program for simulated robotic manipulator arm, or describing with a symbolic motor script the movements of a child or of a motor program for rehabilitation [16].

To make our point about the need for motion knowledge clear, let us look at "music" which is a complex phenomenon similar to motion from many viewpoints. The situation is quite different for music. Indeed, Camurri [5] showed how music notation, as an example of a symbolic representation of a complex phenomenon, was successfully able to discriminate which aspects of complexity to represent explicitly and which to represent implicitly. This successful notation was able to capture the essential structure of music (what in AI terms could be called "music" knowledge). It is easy to notice the abstractness and functionality of the notation, for example the instruments, are not shown in the notation, nor the way in which a performer plays a specific instrument, and the individual style of performance. This adequacy of the music notation symbolism to express the essential structure of music is clearly demonstrated by its ability to survive the advent of computer era.

Music notation can be easily expressed in computer terms and can be used directly to drive computer music synthesizers [3] [17]. For dance and movements, on the contrary, the picture is quite different. Dance and movement notation methods have proliferated, without finding the same success as music notation, and computer techniques directly applied to them do not seem so far to pass the basic test: the ability to generate from the notation a fluent, natural ("convincing") synthetic motion performance [4] [22] [24].

In this paper our principle intention is not to look for yet another movement notation or language for motion, but to tackle motion from a broader prospective. We basically are trying to produce "convincing" animations for a group of articulated bodies such as humans and robots without pressing the animator to become overly involved in the mechanisms of producing the motion. We intend to shift this burden from the animator to the individual articulated bodies through developing an expert animator agent for each member of the collection articulated bodies. These agents have a computational understanding of motion and its semantics in a way that each individual articulated body would handle its motion autonomously. They also have the capabilities to communicate with each other and with the animator. The model allows each agent to have its own behaviour depending on its specific role in the group, duties, areas of responsibilities, etc.

In previous work [14] [15], we have developed a simulation for a multi-legged articulated robot that could be useful in constructing and maintaining structures in space stations such as solar arrays, large multibeam antennas, and space factories. The robot used its legs both for locomotion and for object manipulation. From our review of space literature, we sometimes noted a requirement for more than one robot, since many tasks can only be performed through cooperation of multiple robots. In a sense, we felt that the multi-robot simulation would be a natural extension to our single robot simulation. But, the multiple robot extension brought up two research issues that do not appear in the single robot problem:

(1) The method we used for planning the motion for the single robot was based on the assumption of a static environment, and so it can not be used in the multiple robots case because each of the robots is in a dynamic environment consisting of other moving robots;

(2) The technique we used to animate a single robot's motion was based on calculations of the dynamic equations of motion that were executed on a distributed processor in order to produce the motion in real time. Producing the motion dynamically for the multiple robot case would need much greater processing capability that is currently unavailable.

In order to overcome these two problems and to produce "convincing" animations (as opposed to simulations) for the group of robots without pressing the user to become overly involved in the mechanisms of producing the motion , we have developed an expert animator agent for each robot. Each agent integrates knowledge engineering approaches, namely, object-oriented programming and rule-oriented programming [19] [25] with computer animation approaches. The object-oriented approach plays a key role in the modeling of the robots' inter-relationships, whereas the intelligent functions of each expert animator agent are transparently programmed in rule-oriented programming style. In producing animations for the various robots, each robot is considered to be an object in the environment. It handles its motion and interactions with other robots as well as

with the animator autonomously. To program each expert animator agent intelligently in order to make it adapt to its environment, we adapted the method of production systems.

In order to integrate a rule-oriented approach with an object-oriented approach, the concept of ruleset of "LOOPS," a programming system developed at Xerox PARC, was very instructive [25]. A ruleset is a sort of object which consists of ordered rules with specified control structures for selecting and executing the rules. Each agent keeps several rulesets for performing locomotion, avoiding obstacles, deciding task priorities, etc.

Convincing animations here means that each expert animator agent should be able to maintain the following motion requirements: (1) produce sustained stable locomotion, i.e, maintain its robot orientation, have control over its velocity, and avoid obstacles, (2) choose the most appropriate locomotive skill at any point during the robot's navigation (e.g, walk, trot, climb, etc.), (3) deal intelligently with the environment (Winkless [29] has defined intelligent locomotion as ".. the ability to do appropriate movements under unpredictable conditions"), (4) maintain the robot's static and dynamic stability, (5) ability to combine different locomotion skills (e.g, turning while running), (6) achieve smooth transitions between different locomotive skills, (7) prefer the paths the robot has traveled on before, (8) reduce total energy consumption in executing the robot's missions, (9) perform the robot's task-specific operations elegantly, (10) cooperate with other robots in the environment- either avoid colliding with any of them or cooperating with them in any multi-robot task.

In order for each expert animator agent to satisfy all these requirements, each treats motion in a cognitive framework analogous to co-operation among several motion processes. During motion production, these processes appear or disappear, modify or repeat themselves. Moreover, each expert animator agent has a "motion" knowledge base that provides several levels of sophistication (multiplicity of expressive systems). These levels along with the motion processes are embedded in a formal framework that permits animating the motion at different levels of detail. Each expert animator agent controls its robot's motion at a descriptive level appropriate to the context of its motion. Thus as long as an expert agent determines that its context needs a simple style of movement to produce "convincing" animations, simple techniques will be selected by the agent. As soon as any agent concludes that its context needs more natural, coordinated, task-oriented, and expressive motions, the agent becomes more sophisticated and increases its descriptive level of motion control.

The power of the expert animator agents lies in their generality and "cognitive penetrability" [18] to variations of the basic motion patterns such as uniform trotting, walking over obstacles, overcoming obstacles, etc. In other words the expert agents are able to manage the environmental disturbances without any ad hoc modification to their methodology in motion production. Their formalization power is generic enough to adapt to these disturbances. Moreover, the agents also take care of the robots' interactions with each other. Each expert animator agent optimistically executes each robot's plan without taking into account the existence of other moving robots. Then when two or more robots detect the danger of collision, they negotiate to refine their global path plans in order to avoid collision.

Relying on their "motion" knowledge bases the expert animator agents employ several levels of reasoning in both their negotiations and their answers to the animator's explanation enquiries: (1) Geometric reasoning : both static (e.g, "Am I now on top of obstacle o_i?") and dynamic (e.g, "Can I use the robot's left front leg to reach for the tool t_1 and grasp it?"), (2)Common-sense reasoning (e.g, "Should I switch now to running?", "Should I allow one of the conflicting robots to go first or ask to allow me to do so?").

Actually the multi-robot problem has been investigated in AI under "Distributed Artificial Intelligence (DAI)" where multiple intelligent agents are presented with a complex problem solving situation. Various aspects of DAI research could be found in [6] [7] [8] [26].

The problem has also been investigated in theoritical studies of motion planning for multiple moving objects under "the Piano Movers problem": the problem of planning the motions of several objects among polygonal obstacles. Various aspects and special cases of the research problem could be found in [27]. In this paper we tackle the problem from the graphics animation angle, setting our goal to produce "convincing" animations that satisfy the aforementioned requirements.

In Section 2 the expert animators model is described in the context of our multi-robot space station environment. Section 3 presents a simple experimental system to demonstrate our model. Our conclusions

appear in Section 4.

[2] THE EXPERT ANIMATORS MODEL

Figure 1 shows our multi-robot environment. Each robot has its own behaviour that depends on its specific tasks in the space station. For example we have satellite expert robots, mechanical expert robots, electrical expert robots, etc. Each has its own predefined set of control routines for its specific tasks. All the robots are "physically" alike (see figure 2), they are articulated with four legs each and 18 degrees of freedom. We obtained a crude estimate of the number of degrees of freedom (DOFs) which are needed for free locomotion of each robot by observing that during locomotion it must ideally be possible to control the six DOFs of the body (three translational and three rotational) when it is supported by each of the two alternating sets of legs(at least two legs should be in contact with the station ground all the time to achieve static stability). So one might expect about twelve DOFs to be a minimum for the four-legged robot. If twelve DOFs are taken as a rough estimate, they can be distributed among the robot's four legs as three DOFs each. The robots of figure 1 have three DOFs for each of their four legs: two at the hip (one for elevation and another for lateral movements) and one DOF at the knee.

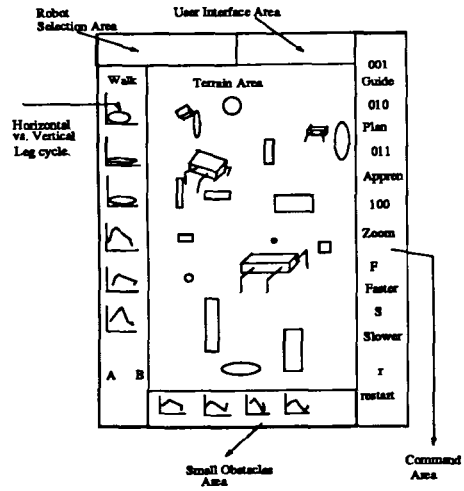

Figure 1: The Multi-Robot Expert Animator System

The robots "live" in a space station that contains obstacles such as blocks, holes, inclines, declines, and rough terrain. Some of the obstacles in the environment are small so the robots do not "see" them until they are navigating close to them. In such cases, the robots have to modify their motions on the fly to avoid or overcome such obstacles upon "perceiving" them. It is important to realize that the robots have no control over what they will encounter in the environment before actual motion execution. Winkless [29] defined intelligent motion as ".. the ability to do appropriate motion under unpredictable conditions". Thus, a preprogrammed motion, highly accurate, productive, precisely measured and well understood in each robot cannot be considered an intelligent one, since a robot has no ability to cope with unpredictable situations and to choose between alternatives. The robots have onboard cameras fixed on top of them. These cameras regularly feed to navigation systems the local obstacles that each robot faces. The traversals of the robots are based on a local navigation strategy that use the on-board camera information [13] [14]. Each robot is capable of four types of tasks: (1) vision-related tasks: scan the surrounding environment- turn the camera 180^0- tilte the camera θ^0 left- align the nearest object to the camera, etc. (2) locomotion tasks: walk, trot, run, turn-left, turn-in-place, stop, etc. (3) task specific operations: grasp, nock, screw, etc. (4) negotiation-related tasks: either to avoid colliding with other robots during navigation, or to co-operate with others in performing multi-robot tasks. The situation we are dealing with here is characterized by the following features: the desired motion trajectories are frequently unknown; the environment is described vaguely (because of the existence of unknown small obstacles); the robots are highly nonlinear, coupled, and redundant. Under these conditions, the robots' expert animator agents are required to

produce purposeful motions ("convincing animations") in real time for all robots.

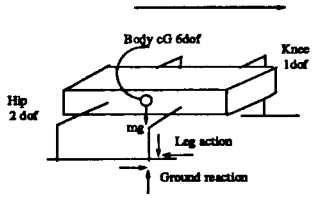

Figure 2: The Four-legged Articulated Robot

Figure 3 shows the internal structure of an expert animator agent. The expert animator agent consists of the following modules: (1)The task planning and execution monitoring module, (2) The agent model, and (3) the agent reasoning module. In the following we describe each in some details.

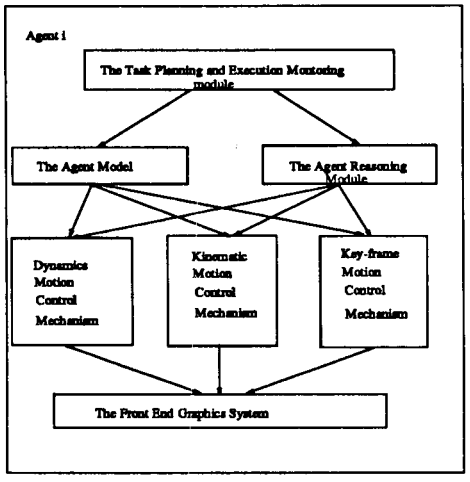

Figure 3: An Expert Animator Agent Internal Structure

[2.1] TASK PLANNING AND EXECUTION MONITORING MODULE

This module generates and executes plans for carrying out tasks that a robot could be performing at any specific time either individually or simultaneously along with other robots. The module consists of a planner and an executor. The planner has knowledge of the large obstacles and objects in the environment; their relative locations and object capabilities. The planner takes an input a task submitted by the animator, or a request from the executor to replan the current task. The planner produces as output a sequence of plan steps to be executed by the different executor routines. For example the task: "Get tool t_i and convey it to robot R_j" could produce the following plan: (1) to the Navigator: "Go to tool box b_k" (2) to the Relative Referencing: "Line-up on the right side along tool box b_k" (3) to the Local Obstacle Avoidance: "Follow the side of the tool box" (4) to the Pattern Matcher: "Identify tool t_i in the tool box- convey its position and orientation (x_1, y_1, θ_1)" (5) to the Trajectory Generator: "Compute a path from a suitable leg/arm to (x_1, y_1, θ_1)" (6) to the Gripper: "Grasp tool t_i at (x_1, y_1, θ_1) using arm AR_k" (7) to the Leg/Arm: "Place tool t_i on top of the robot's body" (8) to the Navigator: "Go to robot R_j" (9) to the Local Obstacle Avoidance: "Follow the side of Robot R_j" (10) to the Trajectory Generator: "Compute a path from a suitable Leg/Arm to R_j's body top" (11) to the Gripper: "Grasp tool t_i from top of the robot's body" (12) to the Leg/Arm: "Place tool t_i on top of robot R_j's body"

Each step in the previous plan is called a plan step. The planning technique that is used by the planers does not concern us here [13]. Upon receiving message of a plan step completion from one of its routines, the executor checks out the success of the step. If the step executes normally the executor proceeds to the next plan step. Otherwise the executor reacts to any abnormal conditions by sending orders to carry out corrective actions

and wait for the next solicitation.

[2.2] AGENT MODEL

The agent model provides several levels of sophistication (a multiplicity of expressive systems) to describe the associated robot's state and its surrounding environment. These levels permit animating the robot's motion at different levels of detail. The agent model contains three levels of descriptions: (1) Conceptual level; (2) Topological level and (3) Dynamics level. Different motion control mechanisms work on these levels. They are, respectively, (1) key-frame motion control mechanism; (2) Kinematics motion control mechanism; (3) Dynamics motion control mechanism. Each expert animator agent views the different levels of descriptions and their manipulation mechanisms as Frame structured data [19] called objects. In object-oriented programming, information and its manipulation mechanisms are put together and represented in the form of objects. Figure 3 shows the different objects that the expert animator deals with. The objects' mechanisms are contained in the agent's reasoning module whereas the robot's descriptive levels are contained in the agent model.

The conceptual level describes the associated robot's capabilities and responsibilities. This includes behaviour characteristics, duties, areas of responsibility, role in a group, etc. Also, general properties of the robot may be expressed here, such as the hands (legs) being used for most grips, the relationship between the size of the object gripped and the capacity of the gripper, etc. The task-specific functions of each robot are represented in a data structure similar to what Turvey [28] called action concepts or what Schank called primitive actions in his Conceptual Dependencies [23] (semantic structure of actions or action verbs in the fields of psychology and linguistics). Any plan step (the output of the Task Planning and Execution Monitoring module) is defined in terms of interrelated component actions, e.g, "reach" for tool t_i, "lift" and "transport" tool t_i above the tool box b_k such that it can be "lowered" into the top opening of the box.

For example Figure 4 represents the plan step for robot R_1 "Use the Screwdriver S_i to screw a screw in the wall". The diagram represents the action of the plan step at its highest node by Agent (x=R_1) screw Object (y=screw) with Implement (z=Screwdriver). The node "screw" includes three semantic subpredicate nodes, each of which stands for a distinct relational concept. An arrow originating from a given node terminates on an entity that is linked through the relation expressed at the arrow's node of origin. Thus, the two predicates "Move" and "Motion" are the arguments for the predicate "Cause", and the labels "event" and "result" indicate the role that "Move" and "Motion" play, respectively, in relation to "Cause". What the Figure represents is that Agent R_1's hand (h_2) movement is an event that causes Implement z (Screwdriver) to move with respect to object y (screw) in a certain spatiotemporal manner. Additionally, the hand and screwdriver are related as implement and object, respectively, through the "Grasp" node; and the screw is located (L) with respect to the tool box b_k through the "Contained In" node.

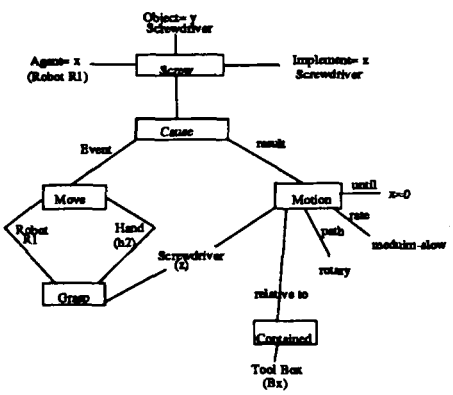

Figure 4: Semantic Representation of the plan step "Use the Screwdriver Si to screw a screw in the wall".

This representation is an abstract specification of the plan step in terms of desired relationships between the robot and objects in the environment. Such relationships are expressed functionally, spatially, and temporally with respect to the action situation. For example, the rotary motion of the screwdriver is specified relative to the wall; however, the precise trajectory of the screwdriver, the shapes of the screwdriver and tool box, and the exact environmental location and orientation of them remain indefinite. What is important is that the representation at this level involves only those spatial characteristics that allow identification of objects and actions as well as action-relevant properties of objects. In a similar way, the temporal parameters of a movement are not defined precisely at this level, but are stated in rather qualitative terms, e.g., the "medium-slow" motion of the screwdriver.

The agent reasoning module (see below) makes use of such conceptual level descriptions of plan steps in two ways: (1)to build key-frames for the associated robot and its surrounding objects to be used as the input for the key-frame motion control mechanism; (2)to provide semantic reasoning through token propagation in the semantic diagram. The purpose of the reasoning here is either to answer the animator's questions about the associated robot's behaviour, or to provide means of high-level negotiations with other robots.

The topological level associates a coordinate frame with each robot's limb and objects in the robot's surrounding environment. Furthermore frames are grouped together to refer to some structures (a robot, an object, etc.). The coordinate frames make us view the robot's actions as streams of variations of some of the mutual relations between the coordinate systems which are due to the stream of motion commands. At this level the expert animator agents view everything as a "forest" of coordinate frames that change over time. By visiting the forest it is possible to express the geometric relations between any two coordinates in the system. This is what in the expert animator agent's reasoning module is called topological reasoning (see later). Figure 5 shows how a coordinate frame is associated with each robot limb and each object in the robot's surrounding environment. In order to perform the plan step explained above ("Use the screwdriver to screw a screw in the wall") the Agent R_1's hand (h_2) frame should overlap with the coordinate frame of the screwdriver and then rotate it with respect to the wall, the screw's frame should overlap with the wall's frame, etc.

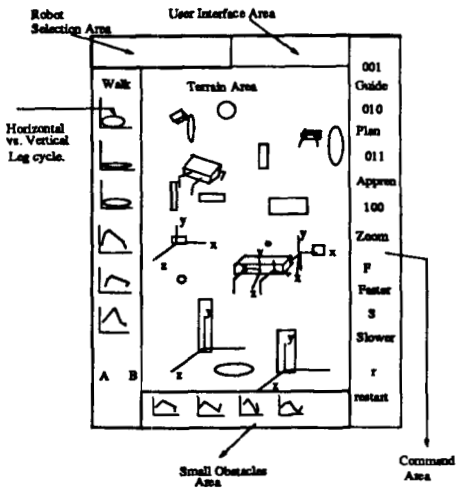

Figure 5:The Geometric level of the agents.

Again the agent reasoning module will make use of such a topological-level description of plan steps in two ways: (1) to provide a framework for the kinematic motion control mechanism (see next section), and (2) to provide topological reasoning either to answer the animator's questions or to interact with other robots.

The dynamics level describes the dynamic properties that are required by the dynamics motion control mechanism in order to perform any plan step dynamically. This includes the masses, locations of centers of masses, moments of inertias, joint spring and damper values, etc.

[2.3] AGENT REASONING MODULE

The agent reasoning module has the responsibility of deciding what are the best mechanisms for producing "convincing" animations for the associated robot in the current motion context. The agent reasoning module takes the planning steps from the Task Planning and Execution Monitoring module and animates the actions at the appropriate sophistication level. The reasoning module evaluates the current context that the associated robot is in (e.g, where the animation camera is, whether the robot is involved in a multi-robot task, whether the robot is out of sight in the current scene, etc.). As long as the agent's reasoning module evaluates that the associated robot's context requires a simple style of movement, simple techniques (e.g, key-frame motion control mechanism) will be selected to drive the associated robot. As soon as the reasoning module determines that the context needs more natural, coordinated, task-oriented, and expressive motions, the reasoning module request the robot to be more sophisticated and increase its level of motion control.

We adapted rule-based programming for describing the various control and decision mechanisms of the agent reasoning module. In order to integrate this rule-oriented approach with the object-oriented organization of the expert animators, rules are organized in rulesets [25]. A ruleset consists of ordered rules with specified control structures for selecting and executing the rules. One such ruleset is the sophistication-level ruleset. Some of its rules are:

IF THE ROBOT IS OFF-SIGHT IN THE NEXT FRAME THEN SET THE CONTROL SWITCHES TO NULL.

IF THE ROBOT IS CLOSE TO THE CAMERA THEN SET THE CONTROL SWITCHES TO DYNAMICS MOTION CONTROL MECHANISM.

IF THE ROBOT IS ENGAGED IN A GROUP TASK THEN SET THE CONTROL SWITCHES TO MIXTURE OF DYNAMICS-KEYFRAME CONTROL MECHANISMS.

IF THERE ARE ANY INTERESTING ACTIONS THEN MOVE THE CAMERA CLOSER TO THE ACTIONS.

The control switches mentioned in the rules are data structure in the reasoning module (see later).

The agent reasoning module has also to provide an explanation-based interface to the animator. Each expert agent should be able to answer the animator's questions about any of the decisions it has taken. For this purpose, the agent reasoning module uses two reasoning mechanisms: (a) Semantic reasoning through token propagation at the conceptual level of the agent model, (b) Geometric reasoning at the geometric level of the agent model.

The Semantic reasoning mechanism answers relationship questions among objects by spreading activation out from each of the objects nodes and seeing where the activations meet [20]. Using this mechanism, it is possible for the agent reasoning module to use a diagram such as the one in Figure 4 to answer questions such as "what is the relationship between the robot R_1 and the screwdriver s_i?". By spreading activation from both the robot R_1 and the screwdriver, the activation meets at the two nodes "Grasp" and "Screw".

The Geometric reasoning mechanism answers geometric types of questions. For example "Are you holding the screwdriver now?", or "Can you reach for place (x_1, y_1) on the wall?". The answer for the first question is "yes"if one of the robot's gripper frame is on top of the screwdriver frame. For the second question the geometric reasoning mechanism will calculate the distance between the frames and access the robot's arm reach capabilities from the agent model.

Finally, the agent reasoning module has the responsibility for navigating its associated robot safetly in the environment. The navigation problem is decomposed into two subproblems: a global path planning problem and a local path planning problem. Each robot's agent reasoning module plans independently its own path for each plan step from its initial location to the final location. No positional constraints introduced by any other robot nor any small obstacles are considered. This is called the global path planning level. At this level the agent reasoning modules use the conventional motion planning solution for the single articulated robot case [15]. However each agent reasoning module will revise its global plan in two situations:

(a) Whenever there is conflict between path plans,i.e, a collision between two or more robots may occur. Robots R_i and R_j are said to be on a collision course between the period t and t+δt if

$$dist(R_i,R_j) \leq (v_i+v_j)\delta t+d$$

where: v_i is the velocity of the robot R_i and v_j is the velocity of robot R_j. d is the safety allowance distance between any two robots. In such cases, domain-specific knowledge describing the robots' current situations is usually used to resolve this conflict. This may include such information as the urgency of one to reach a goal location, the degrees of freedom available for modifying the planned path, the interdependency between the subtasks to be performed by the involved robots, etc. All these data are deduced by the agent reasoning modules from their corresponding agent models. The robots bargain with each other through a process of exchanging knowledge about their situations and suggesting plan revisions. Each agent reasoning module stores some rules for evaluating the descriptions of their relative priorities (this is called the priority ruleset). Example of one such rule is:

IF MY MISSION IS OF PRIORITY=HIGH AND MY TASK STATUS=NEAR-COMPLETION AND NONE OF THE INVOLVED ROBOTS HAS PRIORITY=RUSH THEN ASK FOR THE RIGHT OF THE ROAD.

A robot can start to replan its path and resume its motion if no other robots of higher priority are at a distance shorter than the safe interrobot distance (d). This sequential order of replanning guarantees that the conflict can be resolved. Provided that conflict resolution is not needed very often (i.e. there is enough free space available), the concurrent actions of the robots are only slightly degraded by the sequentiality of the replanning process. Most of the time the robots will be executing their path plans in parallel.

(b) Whenever a robot "perceives" any small obstacle during its locomotion it has to modify its motion on the fly. The reasoning module uses an obstacle avoidance ruleset that identifies the kinds of local obstacles that the robot might face and then uses key-frame motion control mechanism either to avoid the obstacle or to step on top of it. An example of a rule in such ruleset is:

IF LOCAL OBSTACLE= PUMP AHEAD OF LEG_i AND PUMP CHARACTERISTICS ARE (STEEPNESS, FRICTION, SIZE, ETC.)

THEN USE KEY-FRAME MOTION CONTROL MECHANISM TO MODIFY THE MOTION USING $MODIFIER_1$

The reasoning module associates two basic data structures with each robot: Motion Bit Vectors and Control Switches. Following [2] [12] the reasoning module associates a bit vector with every topological frame that exists in the system. A bit vector contains the state of the degrees of freedom that are currently affecting the associated robot's limb or object. Depending on the values of the control switches one or mixture of the motion control mechanisms is responsible for interpreting the goals, constraints, paths, directions (the details of the plan steps) as a series of binary vectors on the various topological frames.

For example if the control switches indicate that the key-frame motion control mechanism should be in control of a robot motion generation, then the bit vectors are interpreted this way: a bit is set in a motion bit vector of a particular limb when a continuous rotational/translational motion about/along the appropriate coordinate axis, is to be used to update the position of this limb in the next frame of motion. The key-frame motion control mechanism will access the semantic representation of the current action plan (see Figure 4) and identify the characteristics of the motion (its path, relative to, rate, until, etc.).

If the control switches indicate that the kinematics motion control mechanism should be in control of robot motion generation, then the bit vectors are interpreted this way: if a bit is set in a motion bit vector of a particular limb then a particular kinematic motion process can effect this limb. There is one bit for each kinematic motion process. Some of the kinematic and dynamic motion processes are rise, fall, jump, swing, hop, lean, pivot, flex a link, bend a link, turn, push, pull, release, grasp, etc.

In general there are two types of kinematic motion process: local kinematic motion processes that effect only the associated limb (i.e. sets bits in only the associated limb's bit vector- e.g. "flex a link"), and global kinematic motion processes that effect several limbs (i.e. sets bits in their bit vectors- e.g. "push"). In the

dynamics motion control mechanism case the interpretation of the bit vectors is identical to the kinematics case except for having dynamics motion processes instead of kinematic ones.

The motion processes are executed on each iteration of the animation. Each degree of freedom in each limb is considered separately. At start of processing for a degree of freedom, its internal rotation (in case of kinematic motion control) or torque/force (in case of dynamic motion control) is set to zero. Then the state bit vector is examined to determine the motion processes that are to be executed. Each motion process uses the current state of the limb, plus its own parameters (stored with the state vector bit) to compute a contribution to the internal rotation (in case of kinematic motion control) or torque/force (in case of dynamic motion control). At the end of this process, new internal rotations or torques/forces will be generated for each limb of the associated robot. These internal rotations or torques/forces will represent the new inputs for the kinematic and dynamic motion control mechanisms.

More details about the motion processes and bit vectors in the particular case of dynamics motion control are treated elsewhere [2].

Similarly, the control switches can be set to indicate that any mixture of the previous motion control mechanisms are participating in the motion production. It is important to mention here that the motion control mechanisms for the articulated bodies (key-frame- kinematics- dynamics) are not mutually exclusive, in the sense that a mixture of kinematics and dynamics has been demonstrated successfully in [9] [10] [30], key-frame and kinematics in [21], key-frame and dynamics in [11]. One of the important features of the proposed model is that it is an open-ended model, in the sense that any new articulated body motion control mechanism could be incorporated into the model. One of the problems with our previous animation system [1] [2] is that the structures of the animated figures and their parameters were hard-coded into the animation routines. In this model, this is replaced by separate agent models that contain complete descriptions of the animated robots and their environment and separate motion control mechanisms that can work on them.

[3] EXPERIMENTAL SYSTEM

An experimental system is under development using an *IRIS** 2400 and several *SUN** workstations [13]. The system is divided into two main components, which are shown in figure 6. The first component, called the front end, is responsible for displaying the robot models and interacting with the animator (see figure 1). This component of the animation system resides on the IRIS and is responsible for displaying and controlling the different expert animator agent's motions. The second component associates a SUN workstation with each expert animator agent. Each agent controls the motion of a particular robot in the system.

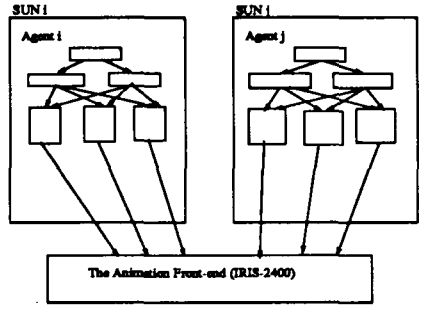

Figure 6:The Experimental System

The two components of the animation system (the IRIS and the various SUNs) communicate by sending packets over an interprocess communications facility (*Unix** sockets: since the workstations are connected by an ethernet). The front end invokes one of the backends when the animator wants to assign a task to the

* IRIS is a trademark of Silicon Graphics, Inc., SUN is a trademark of SUN Microsystems, Inc., and Unix is a trademark of AT&T Bell Laboratories.

corresponding robot. Upon invocation, the expert animator agent of the robot reads the task description and calls the Task Planning and Execution Monitoring module to produce a plan for the locomotion task (In our experimental system we only restricted ourselves to locomotion issues, the manipulation capabilities of the robots were ignored for the purpose of simplifying the experimental system). The reasoning agent module will evaluate the context of the locomotion for the first plan step and will decides on the animation level of the locomotion production (key-frame, kinematic, dynamic, or any mixture) according to its sophistication-level ruleset.

The appropriate motion control mechanism will take over plan step execution and will send a set of packets to the front end. These packets represent the next animation frame to be displayed for the appropriate robot. There is one packet in this set for each limb of the robot, giving its current joint angles. There is also a packet specifying the current position of the body limb of the robot within the environment. At this point the front end responds with one or more packets. These packets are used to inform the expert animator agent of possible collisions with other robots, any small obstacles that are in the robot's way, and the current context of the robot's motion (e.g, the location of the camera with respect to the robot, whether the robot is out of sight in the next scene, etc.). The last packet in this exchange is a Next-frame packet sent from the front end to the expert animator agent. At this point the expert animator agent starts the next frame calculation cycle.

Similar packet exchange take place at the same time between the expert animator agents and the front end. These packet exchanges are synchronized by the front end to ensure that the front end and all the agents are always in step.

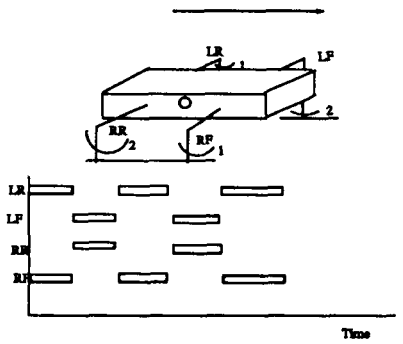

Figure 7:A Robot trotting

The object-oriented approach was quite suitable for the multi-robot animation environment. Viewing each expert animator agent as a distinct object facilitated their separate implementations on the various SUN workstations. In the same manner, the graphics module on the IRIS is also managed according to the object-oriented approach. The interprocess communication packets could be viewed as the messages that are exchanged among the various objects in the system.

The expert animator agents use three locomotion control mechanisms:

(1) key-frame locomotion control mechanism: This is the simplest mechanism to produce a robot's motion. The coordination and synchronization of the joint rotations of the legs are programmed via various locomotion scripts. For example a trotting script is shown in Figure 7.

(2) kinematic locomotion control mechanism: This was based on the work of Girard at Ohio State [9].

(3) dynamic locomotion control mechanism: This was based on our previous work on dynamic locomotion of the single articulated robot [13].

A simple planning algorithm based on A^* was implemented in each expert animator agent with a simple locomotion ruleset that describe the context for using each locomotion skill. An example of one such rule is:

$$IF\ LEG_1[lc_i(X)]\ \&\ LEG_2[lc_i(Y)]\ \&\ LEG_3[lc_i(Z)]\ \&\ LEG_4[lc_i(W)]\ \&\ NAVIGATION\ GOAL = A \rightarrow B$$

$$\&\ ROAD(A,B) = FLAT\ \&\ DISTANCE(A,B) \leq 5\ \&\ INITIAL\text{-}LOC(A) = NS$$

$$\& \; FINAL\text{-}LOC(B) = NS \;\; THEN$$

$$ACTIVATE\ NC\ [lc_1(X'), lc_1(Y'), lc_1(Z'), lc_1(W')]\ UNTIL\ X_E = X\text{-}coordinate\ of\ the\ final\ loc(B)$$

$$\&Y_E = Y\text{-}coordinate\ of\ the\ final\ loc(B)\ WITH\ linear\ speed = [a,b]$$

$$DELETE\ FROM\ WM_1{:}old\ LEG_i\ locs\ ,ADD\ TO\ WM_1{:}$$

$$INITIAL\text{-}LOC(B) = NS\ \ \&LEG_1\ [lc_1(final\ pos)]\&LEG_2\ [lc_1(final\ pos)]$$

$$\&LEG_3\ [lc_1(final\ pos)]\&LEG_4\ [lc_1(final\ pos)]$$

This represents a rule for the usage of the locomotive skill, walking. It provides the conditions under which this skill should be selected to implement the locomotion from A to B. In the action part, it describes how to derive the dynamics routines to implement the skill. Intuitively, this rule says: If the robot's four legs are in some orientation (X,Y,Z,W) within a particular leg cycle (i) (see Figure 1) ,and the goal broadcast by the navigator system is ($A \rightarrow B$) such that the road from A to B is flat and the distance between these two points is less than or equal to 5 units of distance, and both points are located on NS (North-South) direction; then start to drive the NC with the lc_1 leg cycles for all legs such that the legs start from the closest positions to the initial leg settings (X',Y',Z',W') in lc_1. This is in order to facilitate the smooth transitions between different types of gaits. The linear body speed is within the range [a,b], and the stopping conditions are of the destination point B (within some tolerance).

The screen layout for the front end is shown in Figure 1. The display screen is divided into six main sections, called the Terrain Display area, the Robot Selection area, the Locomotion Ruleset area, the Small obstacles Ruleset area, the User Interface area, and the command area. The Terrain Display area display the environment and the various robots at motion. The animator can change the viewing camera position and orientation using the mouse (e.g, zoom-in or zoom-out). As long as the cursor is in the Terrain Display area, the animations of the various robots are displayed. The animator can interrupt the animations by moving the cursor (using the mouse) to any area other than the Terrain Display area.

At the start of the system, the animator moves the cursor to any of the robots and select the robot using the mouse. The robot's name will show on the Robot Selection area of the screen, and the robot color will get changed to indicate that this robot is the current designated robot. At this point the animator can either define a mission for the designated robot, or investigate any enquires about its current status (see below). If the animator wants to assign a mission to the designated robot, s/he should move the cursor to the User Interface area and type in the task for the designated robot (e.g, Goto Place (x_i, z_i)). At this point, when the animator moves the cursor to the Terrain Display area the robot's expert animator agent (on the appropriate SUN) will take over the execution and animation of the task.

At any point the animator can use the Command area to ask a specific robot to move faster; slower; to change the projection view (e.g, orthogonal, prospective); redefine a mission for a particular robot, etc. The Locomotion Ruleset area, the Small Obstacle area, and the User interface area will reflect the current status of any selected robot. The animator can move the cursor and selects any of the robots during the animation. In this case all the internal information of this robot's expert animator agent is available to the animator to manipulate. The status of the designated expert animator agent will be reflected in the various screen areas: (1) The Locomotion Ruleset area will display the current rules that are producing the locomotions of the designated robot. (2) The Small Obstacles Ruleset area will continuously display the current rules that the designated robot's expert agent is using to avoid or overcome the various small obstacles that it is facing as the animation progresses. (3) The User Interface area will display the following information based on the animator requests: (a) the locations of the support legs, (b) the polygon of stability, (c) the predicted polygon of stability when the swinging legs are lowered; (d) the location of the center of gravity, (e) the reachable areas of the legs, (f) the answers to any Reasoning enquiries submitted by the animator.

In the experimental system the sequential ordering of path replanning of section 2.3 was implemented, but no animator explanation (reasoning) capabilities were implemented.

[4] CONCLUSION

The paper describes an advanced expert animator model for animating a group of articulated robots in a three-dimensional environment. The model shifts the burden of human animator involvement with the mechanisms of the robots' motion to various programmed expert animator agents. These agents have computational understanding of motion and its semantics in a way that each handles its robot's behaviour autonomously and communicates with other agents as well as the animator. The design principle of the agents is based on knowledge engineering methods (object-oriented and rule-oriented programming) integrated with computer graphics. The potential of our model is shown by a simple experimental system that was limited to locomotion activities.

REFERENCES

[1] W. Armstrong ,M. Green ,The Dynamics of Articulated Rigid Bodies for Purpose of Animation ,Proceedings of Graphics Interface'85 ,1985

[2] W. Armstrong ,M. Green ,R. Lake Near-Real Time Control of Human Figures Models ,Graphics Interface-86 ,1986

[3] A. Bertoni, G. Haus, G. Mauri, and M. Torelli, "A Mathematical Model for Analyzing and Structuring Musical Texts, Interface 7, 1978.

[4] T. Calvert, J. Chapman, and J. Lindis, "Notation of Dance with Computer Assistance", in New Directions in Dance, D. Taplin, ed., Pergamon Press, Toronto, 1979.

[5] A. Camurri, P. Morasso, V. Tagliasco, and R. Zaccaria, "Dance and Movement Notations", in Human Movement Understanding, North-Holland, 1986.

[6] R. Davis, "Report on the Workshop on Distributed AI", SIGART Newsletter, no. 73, Oct. 1980.

[7] R. Davis, "Report on the Second Workshop on Distributed AI", SIGART Newsletter, no. 80, April 1982.

[8] M. Fehling and L. Erman, "Report on the Third Annual Workshop on Distributed Artificial Intelligence", SIGART Newsletter, no. 84, April 1983.

[9] M. Girard and A. Maciejewski, " Computational Modeling for the Computer Animation of Legged Figures", Computer Graphics 19,3, 1985.

[10] P. Isaacs, M. Cohen, "Controlling Dynamic Simulation with Kinematic Constraints, Behaviour Functions and Inverse Dynamics", Computer Graphics, v. 21, no. 4, July 1987.

[11] D. Lundin, "Simulation", Advanced Computer Animation Tutorial, Siggraph 1986.

[12] T. McMahon, "Mechanics of Locomotion", the International Journal of Robotics Research, 1984, also T. McMahon: Muscles, Reflexes, and Locomotion, Princeton University Press, Princeton N.J., 1984.

[13] A. Mohamed ,A Learning Apprentice System For Locomotion Control of Articulated Bodies ,A Ph.D. thesis in preparation. ,1988

[14] A. Mohamed, W. Armstrong, An Experimental Autonomous Articulated Robot That Can Learn, to appear in the proceeding of the 3rd International Conference on CAD/CAM Robotics & Factories of the Future, Southfield Michigan, August 14-17, 1988.

[15] A. Mohamed, W. Armstrong, A Hybrid Numerical/Knowledge-Based Locomotion Control System for a Multi-legged Manipulator Robot, to appear in the proceeding of the 8th International Workshop on Expert Systems and their Applications, to be held in Avignon, France, June 1-5, 1988.

[16] P. Morasso, V. Tagliasco, "Advances in Psychology: Human Movement Understanding", North-Holland, 1986.

[17] F. Moore, "Introduction to Music Synthesis Using CMUSIC", Technical Report University of San Diego, California, 1982.

[18] Z. Pylyshin, Computation and Cognition, MIT Press, Cambridge, 1984.

[19] D. Robson, "Object-Oriented Software Systems", BYTE, V.6, no.8, 74/86, 1981.

[20] E. Rich, Artificial Intelligence, McGraw-Hill, Reading, 1983.

[21] G.Ridsdale, S. Hewitt, and T. Calvert, "The Interactive Specification of Human Animation", Proc. Graphics Interface '86, Vancouver, 1986.

[22] R. Ryman and B.Singh, "the Benesh Notation Computerized Editor", Proc. Dance in Canada Conf., June 1982.

[23] R. Schank, "Identification of Conceptualizations underlying Natural language", In R.C. Schank & K.M. Colby (Eds), Computer Models of thought and language, San Francisco: Freeman, 1973.

[24] G. Savage and J. Officer, "Choreo: An Interactive Computer Model for Choreography", Proc. of the 5th Man-Machine Comm. Conf., Calgary, Alta, 1977.

[25] M. Stefik, et al, "Rule-Oriented Programming in LOOPS", Symposium on knowledge Engineering and Artificial Intelligence, Information Processing Society of Japan, 65/71, 1985.

[26] R. Smith, "Report on the 1984 Distributed Artificial Intelligence Workshop", AI Magazine, v.6, no.3, Fall 1985.

[27] in Planning, Geometry, and Complexity of Robot Motion, J. Schwatrz, M. Sharir, J. Hopcraft (Eds), Ablex Publishing Corporation, 1987.

[28] M. Turvey, C. Fowler, "Skill Acquisition" An Event Approach with Special Reference to Searching for the Optimum of a function of Many Variables", in G.E. Stelmach (Ed.), Information Processing in Motor Control and Learning, New York, Academic Press, 1978.

[29] N. Winkless and I. Browning, Robots on Your Doorsteps, Robotic Press, Portland, OR, 1978.

[30] J. Wilhelms ,B. Barsky ,Using Dynamic Analysis for the Animation of Articulated Bodies such as Human and Robots ,Proceedings of Graphics Interface'85 ,1985